Исключения

Что такое исключения?

- Исключение — это ошибка, которая возникает во время выполнения программы и нарушает её нормальную работу.
- Исключения помогают делать код более надёжным и устойчивым к ошибкам.

Виды исключений в Python

- Встроенные исключения. Python имеет более 60 типов встроенных исключений, например:

`ZeroDivisionErro`	деление на ноль
`IndexError`	выход за границы списка
`KeyError`	обращение к несуществующему ключу в словаре
`TypeError`	неправильный тип данных
`ValueError`	неверное значение
`FileNotFoundError`	файл не найден

- Пользовательские исключения. Можно создавать свои исключения, унаследовав их от класса Exception:

```
class MyCustomError(Exception):
    pass
raise MyCustomError("Это моя собственная ошибка!")
```

Обработка исключений

- ```
  try:                                                    # базовая обработка
      asdf = 2 / 0
  except:
      print("Возникла ошибка")
  ```

- ```
  try:                                                    # обработка конкретной ошибки
      x = 2 / 0
  except ZeroDivisionError:
      print("Ошибка: деление на ноль!")
  ```

- ```
  try:                                                    # обработка нескольких ошибок
      x = 2 / 0
  except (ZeroDivisionError, IndexError, KeyError):
      print("Какая-то ошибка!")
  ```

- ```
  try:                                                    # обработка всех ошибок
      x = int(“abc")
  except Exception as e:
      print(f"Произошла ошибка: {e}") # Ошибка ValueError
  ```

Конструкция try-except-else-finally

- **`try`** — код, который может вызвать ошибку
  `except` — обработка ошибки
  `else` — выполняется, если ошибки не было
  `finally` — выполняется всегда

- ```
  try:
      num = int(input("Введите число: "))
  except ValueError:
      print("Ошибка: нужно вводить число!")
  else:
      print("Вы ввели число:", num)
  finally:
      print("Эта строка выполнится всегда.")
  ```
  ```

Ручной вызов исключений

- ```
  age = int(input("Введите возраст: "))
  if age < 0:
      raise ValueError("Возраст не может быть отрицательным!")
  ```
  ```

Пользовательские исключения

- ```
  class NegativeNumberError(Exception):
      def __init__(self, message = "Число не может быть отрицательным"):
          self.message = message
          super().__init__(self.message)

  number = int(input("Введите число: "))
    if number < 0:
      raise NegativeNumberError
  ```
  ```

Assert

- оператор для **отладки**, который проверяет, выполняется ли условие. Если условие ложное, Python вызывает исключение AssertionError
- синтаксис: **`assert`** условие, сообщение_об_ошибке
- ```
  age = -2
  assert age > 0, "Возраст должен быть положительным!”

  # Появится ошибка "AssertionError: Возраст должен быть положительным!"
  ```
  ```

Примеры практического использования исключений

- **Работа с файлами и доступом к ресурсам**
  При чтении или записи файлов могут возникнуть ошибки: файл может отсутствовать, быть заблокированным другим процессом или недоступным из-за отсутствия прав. Исключения позволяют обработать эти ситуации и, например, предложить пользователю выбрать другой файл или создать его автоматически.

- **Работа с внешними API и сетевыми запросами**
  При отправке HTTP-запросов сервер может не ответить, вернуть ошибку или прислать некорректные данные. Использование исключений помогает предотвратить крах программы, например, повторно отправить запрос через некоторое время или показать пользователю сообщение о проблеме с подключением.

- **Проверка пользовательского ввода**
  Пользователь может ввести некорректные данные, например, текст вместо числа или дату в неправильном формате. Исключения позволяют обработать ошибку, вывести понятное сообщение и предложить пользователю повторить ввод.

- **Работа с базами данных**
  При выполнении SQL-запросов могут возникнуть ошибки, например, попытка обращения к несуществующей таблице, нарушение ограничений целостности данных или сбой соединения с базой. Исключения помогают корректно завершить транзакцию и избежать повреждения данных.

- **Обработка ошибок в многопоточности и асинхронном программировании**
  В многопоточных и асинхронных приложениях задачи выполняются параллельно, и одна из них может привести к ошибке, не затронув другие. Исключения позволяют поймать такие ошибки и продолжить выполнение программы без полного завершения.